UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/066,833	02/04/2002	Tomasz Konrad Skrzeszewski	NL 010065	4833

7590 08/20/2004

U.S. Philips Corporation
580 White Plains Road
Tarrytown, NY 10591

EXAMINER	
DOLLINGER, BRIAN D	
ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 08/20/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/066,833	SKRZESZEWSKI ET AL.
	Examiner	Art Unit
	Brian D Dollinger	2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *2/4/2002 4/22/2002 7/15/2002*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-13* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-13* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 February 2002* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *7/15/2002*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Brian Dollinger

DETAILED ACTION

1. This office action is in response to documents received on the following dates: 2/4/2002, 4/22/2002, and 7/15/2002.

Priority

2. Receipt is acknowledged of papers submitted under 35 U.S.C. 119(a)-(d), which papers have been placed of record in the file.

Drawings

3. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) that are not mentioned in the description: Figure 1 and 3 elements 144, 162, and 300. Corrected drawing sheets in compliance with 37 CFR 1.121(d), or amendment to the specification to add the reference character(s) in the description in compliance with 37 CFR 1.121(b) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

4. The drawings are objected to as failing to comply with 37 CFR 1.83(a) because the drawings in figures 1, 2, and 3 do not have descriptive labels that clearly describe every feature of the invention. Descriptive labels like "insertion module" or "Execution State" are appropriate to include these labels in the drawings to make them more readable and informative. Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

5. Recommended corrections:

5.1. Figure 1: Elements 120, 140, and 160 should be labeled as their different stages, likewise elements 122, 144, and 162 should be labeled or taken out if not relevant, element 180 should be labeled at the Insertion Module, 142 should be labeled as the Detection Module, element 100 should be labeled as a Data Bus.

5.2. Figure 2: Elements 124 should be labeled as it was labeled in figure 1, elements 284 and 282 should have appropriate names like Storage Device and Insertion Control.

5.3. Figure 3: Element 300 should be labeled as Data Bus, elements 320, 340, and 360 should be labeled their appropriate stages, elements 362a-362e should be labeled as Parallel Execution Stages.

6. In addition to Replacement Sheets containing the corrected drawing figure(s), applicant is required to submit a marked-up copy of each Replacement Sheet including annotations indicating the changes made to the previous version. The marked-up copy must be clearly labeled as "Annotated Marked-up Drawings" and must be presented in the amendment or remarks section that explains the change(s) to the drawings. See 37 CFR 1.121(d). Failure to timely submit the proposed drawing and marked-up copy will result in the abandonment of the application.

Specification

7. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: A method and system for inserting instructions into a pipeline processor to overcome task switching and error stalls and flushes.

8. Applicant is reminded of the proper language and format for an abstract of the disclosure.

The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words. It is important that the abstract not exceed 150 words in length since the space provided for the abstract on the computer tape used by the printer is limited. The form and legal phraseology often used in patent claims, such as "means" and "said," should be avoided. The abstract should describe the disclosure sufficiently to assist readers in deciding whether there is a need for consulting the full patent text for details.

The language should be clear and concise and should not repeat information given in the title. It should avoid using phrases which can be implied, such as, "The disclosure concerns," "The disclosure defined by this invention," "The disclosure describes," etc.

9. The abstract of the disclosure is objected to because "means" is used in lines 2 and 6 and "said" is used in lines 4 and 10. These words must be removed from the abstract and are not appropriate language for use in the abstract. Correction is required. See MPEP § 608.01(b).

10. The abstract of the disclosure is objected to because at the end of the abstract there is a line, "Figure 1". This line needs to be removed because the abstract should only be one paragraph. Correction is required. See MPEP § 608.01(b).

Content of Specification

(a) Title of the Invention: See 37 CFR 1.72(a) and MPEP § 606. The title of the invention should be placed at the top of the first page of

the specification unless the title is provided in an application data sheet. The title of the invention should be brief but technically accurate and descriptive, preferably from two to seven words may not contain more than 500 characters.

(b) Cross-References to Related Applications: See 37 CFR 1.78 and MPEP § 201.11.

(c) Statement Regarding Federally Sponsored Research and Development: See MPEP § 310.

(d) Incorporation-By-Reference Of Material Submitted On a Compact Disc: The specification is required to include an incorporation-by-reference of electronic documents that are to become part of the permanent United States Patent and Trademark Office records in the file of a patent application. See 37 CFR 1.52(e) and MPEP § 608.05. Computer program listings (37 CFR 1.96(c)), "Sequence Listings" (37 CFR 1.821(c)), and tables having more than 50 pages of text were permitted as electronic documents on compact discs beginning on September 8, 2000.

Or alternatively, Reference to a "Microfiche Appendix": See MPEP § 608.05(a). "Microfiche Appendices" were accepted by the Office until March 1, 2001.

(e) Background of the Invention: See MPEP § 608.01(c). The specification should set forth the Background of the Invention in two parts:

(1) Field of the Invention: A statement of the field of art to which the invention pertains. This statement may include a paraphrasing of the applicable U.S. patent classification definitions of the subject matter of the claimed invention. This item may also be titled "Technical Field."

(2) Description of the Related Art including information disclosed under 37 CFR 1.97 and 37 CFR 1.98: A description of the related art known to the applicant and including, if applicable, references to specific related art and problems involved in the prior art which are solved by the applicant's invention. This item may also be titled "Background Art."

(f) Brief Summary of the Invention: See MPEP § 608.01(d). A brief summary or general statement of the invention as set forth in 37 CFR 1.73. The summary is separate and distinct from the abstract

and is directed toward the invention rather than the disclosure as a whole. The summary may point out the advantages of the invention or how it solves problems previously existent in the prior art (and preferably indicated in the Background of the Invention). In chemical cases it should point out in general terms the utility of the invention. If possible, the nature and gist of the invention or the inventive concept should be set forth. Objects of the invention should be treated briefly and only to the extent that they contribute to an understanding of the invention.

(g) Brief Description of the Several Views of the Drawing(s): See MPEP § 608.01(f). A reference to and brief description of the drawing(s) as set forth in 37 CFR 1.74.

(h) Detailed Description of the Invention: See MPEP § 608.01(g). A description of the preferred embodiment(s) of the invention as required in 37 CFR 1.71. The description should be as short and specific as is necessary to describe the invention adequately and accurately. Where elements or groups of elements, compounds, and processes, which are conventional and generally widely known in the field of the invention described and their exact nature or type is not necessary for an understanding and use of the invention by a person skilled in the art, they should not be described in detail. However, where particularly complicated subject matter is involved or where the elements, compounds, or processes may not be commonly or widely known in the field, the specification should refer to another patent or readily available publication which adequately describes the subject matter.

(i) Claim or Claims: See 37 CFR 1.75 and MPEP § 608.01(m). The claim or claims must commence on separate sheet or electronic page (37 CFR 1.52(b)(3)). Where a claim sets forth a plurality of elements or steps, each element or step of the claim should be separated by a line indentation. There may be plural indentations to further segregate subcombinations or related steps. See 37 CFR 1.75 and MPEP § 608.01(i)-(p).

(j) Abstract of the Disclosure: See MPEP § 608.01(f). A brief narrative of the disclosure as a whole in a single paragraph of 150 words or less commencing on a separate sheet following the claims. In an international application which has entered the national stage (37 CFR 1.491(b)), the applicant need not submit an abstract commencing on a separate sheet if an abstract was published with the international application under PCT Article 21. The abstract that appears on the cover page of the pamphlet published by the International Bureau (IB) of the World Intellectual Property

Organization (WIPO) is the abstract that will be used by the USPTO. See MPEP § 1893.03(e).

(k) Sequence Listing, See 37 CFR 1.821-1.825 and MPEP §§ 2421-2431. The requirement for a sequence listing applies to all sequences disclosed in a given application, whether the sequences are claimed or not. See MPEP § 2421.02.

11. The disclosure is objected to because of the following informalities: There are no headings to indicate the section in the specification. Headings are needed for the sections: Background of the Invention, Field of the Invention, Description of the Related Art, Brief Summary of the Invention, Brief Description of the Several Views of the Drawings, and Detailed Description of the Invention. Headings included for the Claims and Abstract of the Disclosure need to be more clear and apparent.

Appropriate corrections are required.

Claim Objections

12. Claims 7, 8, 9, 10, and 12 objected to because of the following informalities: In the claims there are reference numbers that refer to elements in the figures. These reference numbers have not effect on the scope of the claim as stated by 608.01(m) in the MPEP. If it is the applicants desire to have the claims limited by the drawings then the applicant should rewrite the claims stating what limitations should be applied without referring to the drawings. Appropriate correction is required. In this office action no weight will be given to the reference numbers in the claims.

Claim Rejections - 35 USC § 112

13. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

14. Claims 4 and 13 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

15. Claim 4 recites the limitation "said insertion means" in lines 3 and 4. There is insufficient antecedent basis for this limitation in the claim. For the rest of this office action it is assumed that said insertion means refers to forcing an instruction into the pipeline of the processor from claim 1.

16. Claim 4 contains a reference to said instruction B that has been inserted into said first intermediate pipeline stage by said insertion means in lines 2-4 of the claim. Claim 4 is dependent on claim 1 and claim 1 contains an instruction A and an instruction B. Instruction A in claim 1 is inserted into the first intermediate pipeline stage and instruction B is not being inserted and is the instruction that resides in the second intermediate stage of the pipeline. For the rest of this office action It is assumed that instruction B in claim 4 line 2 is meant to be instruction A and that it is a typo.

17. Claim 13 contains a reference to claim 9 but it is unclear if claim 13 is an independent claim or a dependent claim. It is also unclear what limitations in claim 9 effects the code module in claim 13; if the code module in claim 13 is the instruction bundle in claim 9 or constructed from instruction bundles in

claim 9 and/or other instructions. Further it is unclear if the bit pattern in claim 13 is the same bit pattern described in claim 9.

Claim Rejections - 35 USC § 101

18. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

19. Claim 13 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claim 13 is directed to non-statutory subject matter because it claims computer code that is not present on any computer readable medium. To fix this change "A computer program product" to "A computer readable program product".

Claim Rejections - 35 USC § 102

20. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

21. Claims 1-7, 11, and 12 are rejected under 35 U.S.C. 102(b) as being anticipated by Brown et al. (U.S. 5867701).

22. As per claim 1 Brown et al. taught:

    22.1. A method for manipulating an instruction flow (col. 3 lines 19-22).

22.2. In a pipeline of a processor (Figure 1 showing a pipeline diagram and col. 2 lines 64-65).

22.3. Comprising the following steps: detecting a stimulus leading to a disruption of progress of an instruction through a pipeline (Figures 2 and 3 elements 300 showing the module that detects the stimulus that will slowdown the processor and 310 showing the module that inserts instructions into the pipeline by way of the MUX 320 and the decoder 224 and col. 6 lines 50-54).

22.4. On detecting said stimulus, forcing an instruction (col. 6 lines 60-67).

22.5. Required for responding to said stimulus by said processor directly into a first intermediate pipeline stage (Figure 3 elements 224 shows the first intermediate pipeline stage and 310 shows the module that inserts instructions into the pipeline by way of the MUX 320 and the decoder 224).

22.6. Said intermediate stage becoming available as a result of said disruption (col. 6 lines 65-67). The pipeline stage becomes available because the detection hardware detects the stimulus and then changes the flow of the pipeline accordantly to the stimulus detected. This will stall the pipeline and insert one or more instructions into the pipeline behind or in front of the instruction causing the stimulus.

22.7. Characterized in that said stimulus is detected from an instruction type of an instruction B residing in a second intermediate stage of the

pipeline (col. 6 lines 65-67). After an instruction (instruction B) has cause the detections circuits to signal an inserted instruction (instruction A) instruction B moves on to the next stage into the execution stage or it the instruction was already in the execution stage causing a execution error then the it would stay in the execution stage.

22.8. The stages of a pipeline are assumed to be Fetch, Decode, Execute, and Write-back and the intermediate stages are assumed to be Decode and Execute.

23. As per claim 2, an instruction A (the instruction being inserted by the invention) causing the processor to store a processor status on a stack is inherent to Brown et al.'s invention.

23.1. Brown et al. does not explicitly site how exception would be handled by the invention but exceptions are detected and handled by Brown et al.'s invention (faults and invalid opcodes are types of exceptions, Brown et al. col. 6 lines 51-54 and col. 2 lines 45-47). Brown et al.'s invention refers to the Intel family of processors and their instruction sets as being used with Brown et al.'s invention (Brown et al. col. 2 lines 54-56 and specifically the 486 and Pentium processors col. 4 lines 20-24).

23.2. The Intel manual shows how one of ordinary skill in the art at the time of the invention would handle exceptions in Intel architecture. Exceptions cause the processor to store a processor status on a stack (Intel manual vol. 1 page 4-15 lines 1-4 and Intel manual vol. 1 page 4-16

lines 7-8. The contents of the EFLAGS, CS, and EIP registers are the processor status). An exception would cause the insertion unit to insert an instruction into the pipeline to handle the exception and store a processor status on a stack.

23.3. Processor status is assumed to mean any portion of the processor's registers that can be used to resume an interrupted task.

24. As per claim 3, an instruction A (the instruction being inserted by the invention) causing the processor to retrieve processor status from a stack is inherent to Brown et al.'s invention.

24.1. Brown et al. does not explicitly site how exception would be handled by the invention but exceptions are detected and handled by Brown et al.'s invention (faults and invalid opcodes are types of exceptions, Brown et al. col. 6 lines 51-54 and col. 2 lines 45-47). Brown et al.'s invention refers to the Intel family of processors and their instruction sets as being used with Brown et al.'s invention (Brown et al. col. 2 lines 54-56 and specifically the 486 and Pentium processors col. 4 lines 20-24).

24.2. The Intel manual shows how one of ordinary skill in the art at the time of the invention would handle an exception return in Intel architecture. Exception returns cause the processor to retrieve processor status on a stack (Intel manual vol. 1 page 4-16 lines 19-20 and Intel manual vol. 1 page 4-16 lines 26-27. The contents of the EFLAGS, CS, and EIP registers are the processor status and are assumed to be on the

stack because of the way exceptions are called as stated above). An exception return would cause the insertion unit to insert an instruction to retrieve a process status from the stack.

24.3. Processor status is assumed to mean any portion of the processor's registers that can be used to resume an interrupted task.

25. As per claim 4, Brown et al. taught that instruction A (the instruction being inserted by the invention) is an interrupt call that has been inserted into said first intermediate pipeline stage by said insertion means. Brown et al. taught that their invention would receive user input through the code breakpoint circuit figure 5 element 520 such as a keyboard input (Brown et al. col. 9 lines 52-54). Keyboard input is processed by using a hardware interrupt. Input from a keyboard would cause the insertion unit to insert an I/O interrupt into the pipeline.

26. As per claim 5 Brown et al. taught that instruction B (the instruction causing the disturbance) is a programmable instruction causing a pipeline flush.

26.1. Brown et al. does not explicitly site how exception would be handled by the invention but exceptions are detected and handled by Brown et al.'s invention (faults and invalid opcodes are types of exceptions, Brown et al. col. 6 lines 51-54 and col. 2 lines 45-47). Brown et al.'s invention refers to the Intel family of processors and their instruction sets as being used with Brown et al.'s invention (Brown et al. col. 2 lines 54-56 and specifically the 486 and Pentium processors col. 4 lines 20-24).

26.2. The Intel manual shows how one of ordinary skill in the art at the time of the invention would handle exceptions in Intel architecture. An exception would cause a pipeline flush (Intel manual vol. 3 table A-2 on page A-14 row 8 col. 2 and 3).

27. As per claim 6 Brown et al. taught that instruction A (the instruction being inserted by the invention) causes the processor to store a return address on a stack.

27.1. Brown et al. does not explicitly site how exception would be handled by the invention but exceptions are detected and handled by Brown et al.'s invention (faults and invalid opcodes are types of exceptions, Brown et al. col. 6 lines 51-54 and col. 2 lines 45-47). Brown et al.'s invention refers to the Intel family of processors and their instruction sets as being used with Brown et al.'s invention (Brown et al. col. 2 lines 54-56 and specifically the 486 and Pentium processors col. 4 lines 20-24).

27.2. The Intel manual shows how one of ordinary skill in the art at the time of the invention would handle exceptions in Intel architecture. An exception would cause a pipeline flush (Intel manual vol. 3 table A-2 on page A-14 row 8 col. 2 and 3) and it would store a return address on a stack by storing the CS and EIP registers (Intel manual vol. 3 page 5-15 lines 20-23) and exception.

28. As per claim 7 Brown et al. taught:

28.1. A system for manipulating an instruction flow (col. 3 lines 19-22).

28.2. Comprising: a processor having a processing pipeline (Figure 1 showing a pipeline diagram and col. 2 lines 64-65).

28.3. Detection means for detecting a stimulus leading to a disruption of the progress of an instruction through said pipeline (Figures 2 and 3 elements 300, showing the module that detects the stimulus that will slowdown the processor and 310 showing the module that inserts instructions into the pipeline by way of the MUX 320 and the decoder 224 and col. 6 lines 50-54).

28.4. Insertion means, responsive to said detection means (Figure 3 element 310 as indicated the module that inserts instructions in to the pipeline).

28.5. For forcing an instruction A directly into a first intermediate pipeline stage (Figure 3 elements 224 shows the first intermediate pipeline stage and 310 shows the module that inserts instructions into the pipeline by way of the MUX 320 and the decoder 224). The insertion module 221 inserts the instructions into the decode stage of the pipeline this being the first intermediate pipeline stage.

28.6. Said stage becoming available as a result of said disruption, characterized in that said stimulus is detectable from an instruction type of an instruction B (Figure 3 element 300 shows the detection circuits and col. 6 lines 51-54). The pipeline stage becomes available because the detection hardware detects the stimulus and then changes the flow of the pipeline accordantly to the stimulus detected. This will stall the pipeline

and insert one or more instructions into the pipeline behind or in front of the instruction causing the stimulus.

28.7. Residing in a second intermediate stage of the pipeline (col. 6 lines 65-67). After an instruction (instruction B) has cause the detections circuits to signal an inserted instruction (instruction A) instruction B moves on to the next stage into the execution stage or it the instruction was already in the execution stage causing a execution error then the it would stay in the execution stage.

28.8. The stages of a pipeline are assumed to be Fetch, Decode, Execute, and Write-back and the intermediate stages are assumed to be Decode and Execute.

29. As per claim 11 Brown et al. taught that the instruction to be forced into a pipeline by said insertion means is present in the system in a hard-coded manner (col. 7 lines 36-40 and 49-51).

30. As per claim 12 Brown et al. taught that the instruction to be forced into a pipeline by said insertion means is stored in a data storage device (col. 7 lines 37-40 and 49-51).

Claim Rejections - 35 USC § 103

31. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which

said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

32. Claims 8-10 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Brown et al. as applied to claim 7 above, and further in view of Miller (U.S. 6081884).

33. As per claim 8 Brown et al. taught:

33.1. Pipeline comprises a plurality of execute stages for executing the plurality of instructions of said instruction bundle in a parallel fashion (Figure 2 element 280 and col. 6 lines 31-37).

33.2. Detections means precedes the plurality of execute stages (Figure 2 element 221 and 280).

33.3. However Brown et al. does not teach that Instruction B (the instruction causing the disturbance) is an element of an instruction bundle comprising a plurality of instructions.

33.4. Brown et al.'s invention is not a VLIW or LIW architecture, it is an X86 architecture. If Brown et al.'s invention was a VLIW or LIW architecture then the detection unit would detect a disturbance caused by one of the instructions in the bundle.

33.5. Miller taught the use of the X86 instruction set in a LIW format (col. 1 lines 60-63 and Figure 2 showing multiple instructions bundled together to make one long instruction) in which more then one instruction is contained in each instruction bundle (col. 1 lines 67, col. 2 lines 1-3, and Figure 2). This was done to gain performance by exploiting more

parallelism (col. 2 lines 7-9) while utilizing the vast amount of code already written for the X86 architecture (col. 2 lines 30-35).

33.6. The gain in performance through exploiting more parallelism using a LIW format along with the large amount of code already written for X86 processors would have motivated one skilled in the art at the time of the invention to combine Brown et al.'s invention with the teachings of Miller to utilize instruction bundles containing multiple instructions in the LIW architecture.

34. As per claim 9 Brown et al. taught the detection means is arranged to evaluate a bit pattern attached to the instruction, with the combination of Brown et al. with Miller from claim 8 the instruction will become an instruction bundle (col. 2 lines 55-56) and a bit pattern marking the presence of said instruction type amongst said plurality of instructions (col. 2 lines 55-56). Brown et al. sites the use of special instruction prefixes that can be used to identify instructions that would be detected this prefix would inherently be directed to specific instruction in the instruction bundle.

35. As per claim 10 Brown did not teach that the instruction bundle in claim 8 was a Very Long Instruction Word (VLIW).

35.1. Miller taught the use of the X86 instruction set in a LIW format (col. 1 lines 60-63 and Figure 2 showing multiple instructions bundled together to make one long instruction) in which more then one instruction is contained in each instruction bundle (col. 1 lines 67, col. 2 lines 1-3, and Figure 2). This was done to gain performance by exploiting more

parallelism (col. 2 lines 7-9) while utilizing the vast amount of code already written for the X86 architecture (col. 2 lines 30-35).

35.2. The gain in performance through exploiting more parallelism using a LIW format along with the large amount of code already written for X86 processors would have motivated one skilled in the art at the time of the invention to combine Brown et al.'s invention with the teachings of Miller to utilize instruction bundles containing multiple instructions in the LIW architecture.

35.3. The LIW format used in Miller is the same instruction format that Skrzeszewski et al. uses in their application. LIW and VLIW instruction formats refer to the same type of architecture in this case.

36. As per claim 13 Brown et al. taught:

36.1. A computer program product comprising a code module for execution by the system of claim 9 (the combination of Brown et al. and Miller's system inherently has code that runs on the system).

36.2. Characterized in that said code module comprises an instruction extended with a bit pattern (col. 2 lines 55-56, special instruction prefixes).

36.3. Said bit pattern making said instruction recognizable to the detection means of one of said systems (col. 8 lines 40-42).

Conclusion

37. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Brian D Dollinger whose telephone number is (703) 305-8978 or (571) 272-4164. The examiner can normally be reached on 8-4:30 M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703)305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Brian D Dollinger
Examiner
Art Unit 2183

BDD

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

2183
BT
7-19-02

INFORMATION DISCLOSURE STATEMENT TRANSMITTAL

To Commissioner For Patents
Enclosed herewith is a Form PTO-1449, required copies of documents listed thereon, and a concise explanation of their relevance is described below or enclosed herewith per 37 CFR 1.97.

Applicati n Number	10/066,833
Filing Date	February 4, 2002
First Named Inventor	Tomasz K. Skrzeszewski et al
Group Art Unit	2183
Examiner Name	
Attorney Docket Number	NL 010065

These documents may be relevant in that they have been (check one):

☐ considered in drafting the specification of the above referenced application;

☐ cited in the specification of the above-referenced application;

☐ previously submitted or cited in U.S. patent application(s) which are relied on for an earlier effective filing date under 35 U.S.C. 120 (no copies required);

☒ cited as an "X" or "Y" document in a foreign Patent Office search report in a foreign counterpart application, a copy of which report is also enclosed; or

☐ otherwise a concise explanation of the relevance of each document, as understood by the individual designated in §1.56(c) most knowledgeable about the contents, is append hereto.

if the date of this IDS may be after the date of a final Office Action - per §1.97(e) check one:

☐ I hereby certify that these documents were first cited in any communication with a foreign Patent Office for a counterpart foreign application not more than three (3) months ago; or

☐ I hereby certify that not one of these documents was cited in any communication with a foreign Patent Office in a counterpart foreign application, nor was any known to any individual designated in §1.56(c) more than three (3) months ago.

Please charge any required fee under § 1.17(p) or any other required fee (except the issue fee) to Account No. 14-1270.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Name (Print Type)	Michael E. Marion	Registration No. (Attorney/Agent)	32,266
Signature	Michael E. Marion	Date	July 12, 2002

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this is being deposited with the U.S. Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner For Patents, Washington, DC 20231, or facsimile transmitted to the U.S. Patent and Trademark Office tel# : _____ on the date below:

Name (Print Type)	Patti DeMichele		
Signature	Patti DeMichele	Date	July 12, 2002

Form PTO-1449 U.S. DEPARTMENT OF COMMERCE (REV. 7-80) PATENT AND TRADEMARK OFFICE	Atty. Docket No. NB 010065	Serial No. 10/066,833
	Applicant TOMASZ K. SKRZESZEWSKI ET AL	
	Filing Date February 4, 2002	Group 2183

INFORMATION DISCLOSURE CITATION
(Use several sheets if necessary)

U.S. PATENT DOCUMENTS

Ex. Int		Document Number	Date	Name	Class	Sub-class	Filing Date If Approp.
BCC	AA	5 8 6 7 7 0 1	2/2/99	Brown et al	395	598	9/24/97
	AB						
	AC						
	AD						
	AE						

FOREIGN PATENT DOCUMENTS

		Document Number	Date	Country	Class	Sub-class	Trans. Yes	Trans. no
BDD	AF	0 4 6 3 9 6 6 A 2	1/2/92	Europe			X	
	AG							
	AH							
	AI							
	AJ							
	AK							

OTHER (Including Author, Title, Date, Pertinent Pages, Etc.)

	AL	
	AM	
	AN	

Examiner Brian Dellinger Brian Dellinger Date Considered 8/10/04

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP Draw line through citation if not in conformance and not considered. Include a copy this form with next communication to applicant.

INFORMATION DISCLOSURE STATEMENT TRANSMITTAL

To Commissioner For Patents
Enclosed herewith is a Form PTO-1449, required copies of documents listed thereon, and a concise explanation of their relevance is described below or enclosed herewith per 37 CFR 1.97.

Application Number	
Filing Date	CONCURRENTLY
First Named Inventor	TOMASZ K. SKRZESZEWSKI ET AL
Group Art Unit	
Examiner Name	
Attorney Docket Number	NL 010065

These documents may be relevant in that they have been:

[x] considered in drafting the specification of the above-Referenced application;

[x] cited in the specification of the above-referenced Application;

[ ] Previously submitted or cited in U.S. patent application(s) _____ which are relied on for an earlier effective filing date under 35 U.S.C. 120 (no copy required); or

[ ] cited as an "X" or "Y" document in a foreign Patent Office search report on a foreign counter part application, a copy of which report is also enclosed;

[ ] I hereby certify that these documents were first cited in any communication with a foreign Patent Office for a counterpart foreign application not more than three (3) months ago;

[ ] Otherwise a concise explanation of the relevance of each document is append hereto.

[ ] I hereby certify that not one of these documents was cited in any communication with a foreign Patent Office nor was any known to any individual designated in §1.56(c) more than three (3) m months ago.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQURIED

Name (Print Type)	Michael E. Marion	*Registration No. (Attorney/Agent)*	32,266
Signature		*Date*	February 5, 2002

S:\ga\PRIORART.DOC

Form PTO-1449 (REV. 7-80) U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

Atty. Docket No. NL 010065	Serial No
Applicant TOMASZ K. SKRZESZEWSKI ET AL	
Filing Date Concurrently	Group

J1050 U.S. PTO 10/066993 02/04/02

INFORMATION DISCLOSURE CITATION

(Use several sheets if necessary)

U.S. PATENT DOCUMENTS

Ex. Int		Document Number	Date	Name	Class	Sub-class	Filing Date If Approp.
	AA						
	AB						
	AC						
	AD						
	AE						

FOREIGN PATENT DOCUMENTS

		Document Number	Date	Country	Class	Sub-class	Trans. Yes	Trans. no
RD	AF	9 9 1 8 4 9 7 A 1	4/15/99	WORLD			X	
	AG							
	AH							
	AI							
	AJ							

OTHER (Including Author, Title, Date, Pertinent Pages, Etc.)

	AK	
	AL	
	AM	

Examiner Brian Dollinger Date Considered 8/10/04

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP Draw line through citation if not in conformance and not considered. Include a copy this form with next communication to applicant.

Notice of References Cited	Application/Control No. 10/066,833	Applicant(s)/Patent Under Reexamination SKRZESZEWSKI ET AL.	
	Examiner Brian D Dollinger	Art Unit 2183	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-5,867,701	02-1999	Brown et al.	712/248
	B	US-6,081,884	06-2000	Miller, Paul K.	712/204
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Intel Architecture Software Developer's Manual, 1999 Vol. 1 Pages 4-11, 4-14 to 4-16, and 6-36 Vol. 3 Pages 5-4, 5-15, 6-10 to 6-12, 16-2, A-14
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Intel Architecture Software Developer's Manual

Volume 1: Basic Architecture

NOTE: The *Intel Architecture Software Developer's Manual* consists of three volumes: *Basic Architecture*, Order Number 243190; *Instruction Set Reference*, Order Number 243191; and the *System Programming Guide*, Order Number 243192.
Please refer to all three volumes when evaluating your design needs.

1999

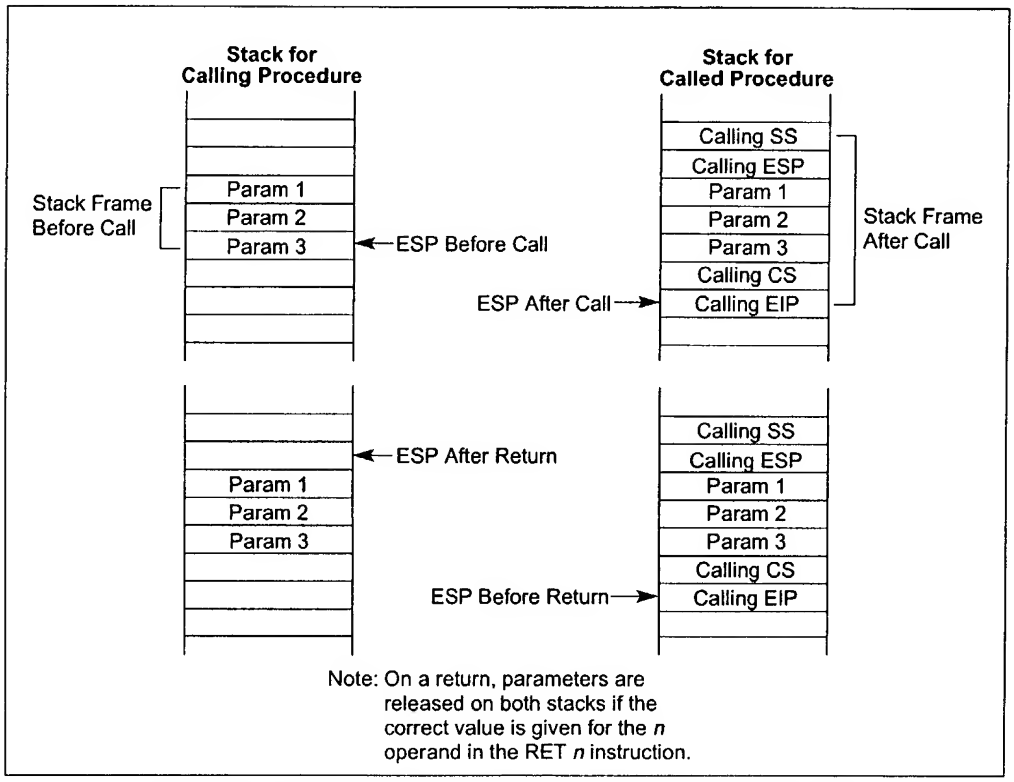

Figure 4-4. Stack Switch on a Call to a Different Privilege Level

Refer to Chapter 4, *Protection* of the *Intel Architecture Software Developer's Manual, Volume 3*, for detailed information on calls to privileged levels and the call gate descriptor.

4.4. INTERRUPTS AND EXCEPTIONS

The processor provides two mechanisms for interrupting program execution: interrupts and exceptions:

- An **interrupt** is an asynchronous event that is typically triggered by an I/O device.
- An **exception** is a synchronous event that is generated when the processor detects one or more predefined conditions while executing an instruction. The IA specifies three classes of exceptions: faults, traps, and aborts.

The processor responds to interrupts and exceptions in essentially the same way. When an interrupt or exception is signaled, the processor halts execution of the current program or task and switches to a handler procedure that has been written specifically to handle the interrupt or exception condition. The processor accesses the handler procedure through an entry in the inter-

Table 4-1. Exceptions and Interrupts

Vector No.	Mnemonic	Description	Source
0	#DE	Divide Error	DIV and IDIV instructions.
1	#DB	Debug	Any code or data reference.
2		NMI Interrupt	Non-maskable external interrupt.
3	#BP	Breakpoint	INT 3 instruction.
4	#OF	Overflow	INTO instruction.
5	#BR	BOUND Range Exceeded	BOUND instruction.
6	#UD	Invalid Opcode (UnDefined Opcode)	UD2 instruction or reserved opcode.[1]
7	#NM	Device Not Available (No Math Coprocessor)	Floating-point or WAIT/FWAIT instruction.
8	#DF	Double Fault	Any instruction that can generate an exception, an NMI, or an INTR.
9		CoProcessor Segment Overrun (reserved)	Floating-point instruction.[2]
10	#TS	Invalid TSS	Task switch or TSS access.
11	#NP	Segment Not Present	Loading segment registers or accessing system segments.
12	#SS	Stack Segment Fault	Stack operations and SS register loads.
13	#GP	General Protection	Any memory reference and other protection checks.
14	#PF	Page Fault	Any memory reference.
15		(Intel reserved. Do not use.)	
16	#MF	Floating-Point Error (Math Fault)	Floating-point or WAIT/FWAIT instruction.
17	#AC	Alignment Check	Any data reference in memory.[3]
18	#MC	Machine Check	Error codes (if any) and source are model dependent.[4]
19	#XF	Streaming SIMD Extensions	SIMD floating-point numeric exceptions.[5]
20-31		(Intel reserved. Do not use.)	
32-255		Maskable Interrupts	External interrupt from INTR pin or INT *n* instruction.

1. The UD2 instruction was introduced in the Pentium® Pro processor.
2. IA processors after the Intel386™ processor do not generate this exception.
3. This exception was introduced in the Intel486™ processor.
4. This exception was introduced in the Pentium® processor and enhanced in the Pentium® Pro processor.
5. This exception was introduced in the Pentium® III processor.

If no stack switch occurs, the processor does the following when calling an interrupt or exception handler (refer to Figure 4-5):

1. Pushes the current contents of the EFLAGS, CS, and EIP registers (in that order) on the stack.
2. Pushes an error code (if appropriate) on the stack.
3. Loads the segment selector for the new code segment and the new instruction pointer (from the interrupt gate or trap gate) into the CS and EIP registers, respectively.
4. If the call is through an interrupt gate, clears the IF flag in the EFLAGS register.
5. Begins execution of the handler procedure at the new privilege level.

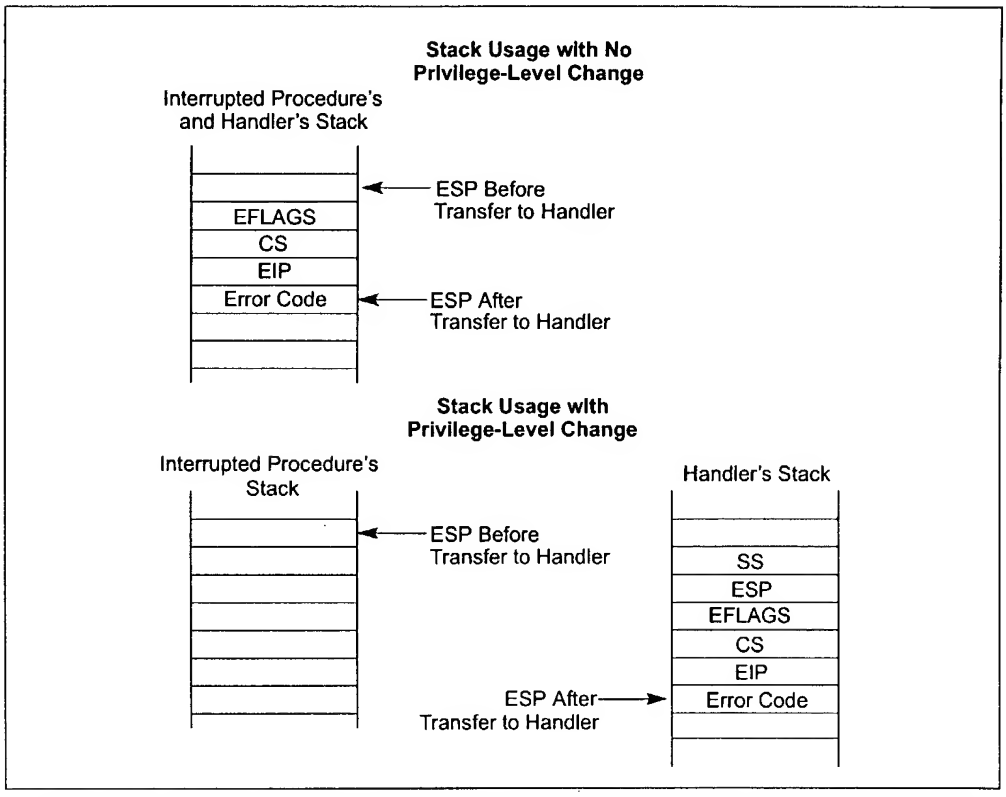

Figure 4-5. Stack Usage on Transfers to Interrupt and Exception Handling Routines

If a stack switch does occur, the processor does the following:

1. Temporarily saves (internally) the current contents of the SS, ESP, EFLAGS, CS, and EIP registers.
2. Loads the segment selector and stack pointer for the new stack (that is, the stack for the privilege level being called) from the TSS into the SS and ESP registers and switches to the new stack.
3. Pushes the temporarily saved SS, ESP, EFLAGS, CS, and EIP values for the interrupted procedure's stack onto the new stack.
4. Pushes an error code on the new stack (if appropriate).
5. Loads the segment selector for the new code segment and the new instruction pointer (from the interrupt gate or trap gate) into the CS and EIP registers, respectively.
6. If the call is through an interrupt gate, clears the IF flag in the EFLAGS register.
7. Begins execution of the handler procedure at the new privilege level.

A return from an interrupt or exception handler is initiated with the IRET instruction. The IRET instruction is similar to the far RET instruction, except that it also restores the contents of the EFLAGS register for the interrupted procedure:

When executing a return from an interrupt or exception handler from the same privilege level as the interrupted procedure, the processor performs these actions:

1. Restores the CS and EIP registers to their values prior to the interrupt or exception.
2. Restores the EFLAGS register.
3. Increments the stack pointer appropriately
4. Resumes execution of the interrupted procedure.

When executing a return from an interrupt or exception handler from a different privilege level than the interrupted procedure, the processor performs these actions:

1. Performs a privilege check.
2. Restores the CS and EIP registers to their values prior to the interrupt or exception.
3. Restores the EFLAGS register.
4. Restores the SS and ESP registers to their values prior to the interrupt or exception, resulting in a stack switch back to the stack of the interrupted procedure.
5. Resumes execution of the interrupted procedure.

6.9.1.2. CALL AND RETURN INSTRUCTIONS

The CALL (call procedure) and RET (return from procedure) instructions allow a jump from one procedure (or subroutine) to another and a subsequent jump back (return) to the calling procedure.

The CALL instruction transfers program control from the current (or calling procedure) to another procedure (the called procedure). To allow a subsequent return to the calling procedure, the CALL instruction saves the current contents of the EIP register on the stack before jumping to the called procedure. The EIP register (prior to transferring program control) contains the address of the instruction following the CALL instruction. When this address is pushed on the stack, it is referred to as the **return instruction pointer** or **return address.**

The address of the called procedure (the address of the first instruction in the procedure being jumped to) is specified in a CALL instruction the same way as it is in a JMP instruction (refer to Section 6.9.1.1., "Jump Instruction"). The address can be specified as a relative address or an absolute address. If an absolute address is specified, it can be either a near or a far pointer.

The RET instruction transfers program control from the procedure currently being executed (the called procedure) back to the procedure that called it (the calling procedure). Transfer of control is accomplished by copying the return instruction pointer from the stack into the EIP register. Program execution then continues with the instruction pointed to by the EIP register.

The RET instruction has an optional operand, the value of which is added to the contents of the ESP register as part of the return operation. This operand allows the stack pointer to be incremented to remove parameters from the stack that were pushed on the stack by the calling procedure.

Refer to Section 4.3., "Calling Procedures Using CALL and RET" in Chapter 4, *Procedure Calls, Interrupts, and Exceptions* for more information on the mechanics of making procedure calls with the CALL and RET instructions.

6.9.1.3. RETURN FROM INTERRUPT INSTRUCTION

When the processor services an interrupt, it performs an implicit call to an interrupt-handling procedure. The IRET (return from interrupt) instruction returns program control from an interrupt handler to the interrupted procedure (that is, the procedure that was executing when the interrupt occurred). The IRET instruction performs a similar operation to the RET instruction (refer to Section 6.9.1.2., "Call and Return Instructions") except that it also restores the EFLAGS register from the stack. The contents of the EFLAGS register are automatically stored on the stack along with the return instruction pointer when the processor services an interrupt.

6.9.2. Conditional Transfer Instructions

The conditional transfer instructions execute jumps or loops that transfer program control to another instruction in the instruction stream if specified conditions are met. The conditions for control transfer are specified with a set of condition codes that define various states of the status flags (CF, ZF, OF, PF, and SF) in the EFLAGS register.

Intel Architecture Software Developer's Manual

Volume 3: System Programming

NOTE: The *Intel Architecture Software Developer's Manual* consists of three volumes: *Basic Architecture*, Order Number 243190; *Instruction Set Reference,* Order Number 243191; and the *System Programming Guide,* Order Number 243192.
Please refer to all three volumes when evaluating your design needs.

1999

5.1.2.3. MACHINE-CHECK EXCEPTIONS

The P6 family and Pentium® processors provide both internal and external machine-check mechanisms for checking the operation of the internal chip hardware and bus transactions. These mechanisms constitute extended (implementation dependent) exception mechanisms. When a machine-check error is detected, the processor signals a machine-check exception (vector 18) and returns an error code. Refer to "Interrupt 18—Machine Check Exception (#MC)" at the end of this chapter and Chapter 13, *Machine-Check Architecture*, for a detailed description of the machine-check mechanism.

5.2. EXCEPTION AND INTERRUPT VECTORS

The processor associates an identification number, called a **vector**, with each exception and interrupt. Table 5-1 shows the assignment of exception and interrupt vectors. This table also gives the exception type for each vector, indicates whether an error code is saved on the stack for an exception, and gives the source of the exception or interrupt.

The vectors in the range 0 through 31 are assigned to the exceptions and the NMI interrupt. Not all of these vectors are currently used by the processor. Unassigned vectors in this range are reserved for possible future uses. **Do not use the reserved vectors.**

The vectors in the range 32 to 255 are designated as user-defined interrupts. These interrupts are not reserved by the Intel Architecture and are generally assigned to external I/O devices and to permit them to signal the processor through one of the external hardware interrupt mechanisms described in Section 5.1.1., "Sources of Interrupts"

5.3. EXCEPTION CLASSIFICATIONS

Exceptions are classified as **faults**, **traps**, or **aborts** depending on the way they are reported and whether the instruction that caused the exception can be restarted with no loss of program or task continuity.

Faults — A fault is an exception that can generally be corrected and that, once corrected, allows the program to be restarted with no loss of continuity. When a fault is reported, the processor restores the machine state to the state prior to the beginning of execution of the faulting instruction. The return address (saved contents of the CS and EIP registers) for the fault handler points to the faulting instruction, rather than the instruction following the faulting instruction.

Note: There are a small subset of exceptions that are normally reported as faults, but under architectural corner cases, they are not restartable and some processor context will be lost. An example of these cases is the execution of the POPAD instruction where the stack frame crosses over the the end of the stack segment. The exception handler will see that the CS:EIP has been restored as if the POPAD instruction had not executed however internal processor state (general purpose registers) will have been modified. These corner cases are

code segment. These gates differ in the way the processor handles the IF flag in the EFLAGS register (refer to Section 5.10.1.2., "Flag Usage By Exception- or Interrupt-Handler Procedure").

5.10. EXCEPTION AND INTERRUPT HANDLING

The processor handles calls to exception- and interrupt-handlers similar to the way it handles calls with a CALL instruction to a procedure or a task. When responding to an exception or interrupt, the processor uses the exception or interrupt vector as an index to a descriptor in the IDT. If the index points to an interrupt gate or trap gate, the processor calls the exception or interrupt handler in a manner similar to a CALL to a call gate (refer to Section 4.8.2., "Gate Descriptors" through Section 4.8.6., "Returning from a Called Procedure" in Chapter 4, *Protection*). If index points to a task gate, the processor executes a task switch to the exception- or interrupt-handler task in a manner similar to a CALL to a task gate (refer to Section 6.3., "Task Switching" in Chapter 6, *Task Management*).

5.10.1. Exception- or Interrupt-Handler Procedures

An interrupt gate or trap gate references an exception- or interrupt-handler procedure that runs in the context of the currently executing task (refer to Figure 5-3). The segment selector for the gate points to a segment descriptor for an executable code segment in either the GDT or the current LDT. The offset field of the gate descriptor points to the beginning of the exception- or interrupt-handling procedure.

When the processor performs a call to the exception- or interrupt-handler procedure, it saves the current states of the EFLAGS register, CS register, and EIP register on the stack (refer to Figure 5-4). (The CS and EIP registers provide a return instruction pointer for the handler.) If an exception causes an error code to be saved, it is pushed on the stack after the EIP value.

If the handler procedure is going to be executed at the same privilege level as the interrupted procedure, the handler uses the current stack.

If the handler procedure is going to be executed at a numerically lower privilege level, a stack switch occurs. When a stack switch occurs, a stack pointer for the stack to be returned to is also saved on the stack. (The SS and ESP registers provide a return stack pointer for the handler.) The segment selector and stack pointer for the stack to be used by the handler is obtained from the TSS for the currently executing task. The processor copies the EFLAGS, SS, ESP, CS, EIP, and error code information from the interrupted procedure's stack to the handler's stack.

To return from an exception- or interrupt-handler procedure, the handler must use the IRET (or IRETD) instruction. The IRET instruction is similar to the RET instruction except that it restores the saved flags into the EFLAGS register. The IOPL field of the EFLAGS register is restored only if the CPL is 0. The IF flag is changed only if the CPL is less than or equal to the IOPL. Refer to "IRET/IRETD—Interrupt Return" in Chapter 3 of the *Intel Architecture Software Developer's Manual, Volume 2*, for the complete operation performed by the IRET instruction.

If a stack switch occurred when calling the handler procedure, the IRET instruction switches back to the interrupted procedure's stack on the return.

A task can be accessed either through a task-gate descriptor or a TSS descriptor. Both of these structures are provided to satisfy the following needs:

- The need for a task to have only one busy flag. Because the busy flag for a task is stored in the TSS descriptor, each task should have only one TSS descriptor. There may, however, be several task gates that reference the same TSS descriptor.
- The need to provide selective access to tasks. Task gates fill this need, because they can reside in an LDT and can have a DPL that is different from the TSS descriptor's DPL. A program or procedure that does not have sufficient privilege to access the TSS descriptor for a task in the GDT (which usually has a DPL of 0) may be allowed access to the task through a task gate with a higher DPL. Task gates give the operating system greater latitude for limiting access to specific tasks.
- The need for an interrupt or exception to be handled by an independent task. Task gates may also reside in the IDT, which allows interrupts and exceptions to be handled by handler tasks. When an interrupt or exception vector points to a task gate, the processor switches to the specified task.

Figure 6-6 illustrates how a task gate in an LDT, a task gate in the GDT, and a task gate in the IDT can all point to the same task.

6.3. TASK SWITCHING

The processor transfers execution to another task in any of four cases:

- The current program, task, or procedure executes a JMP or CALL instruction to a TSS descriptor in the GDT.
- The current program, task, or procedure executes a JMP or CALL instruction to a task-gate descriptor in the GDT or the current LDT.
- An interrupt or exception vector points to a task-gate descriptor in the IDT.
- The current task executes an IRET when the NT flag in the EFLAGS register is set.

The JMP, CALL, and IRET instructions, as well as interrupts and exceptions, are all generalized mechanisms for redirecting a program. The referencing of a TSS descriptor or a task gate (when calling or jumping to a task) or the state of the NT flag (when executing an IRET instruction) determines whether a task switch occurs.

The processor performs the following operations when switching to a new task:

1. Obtains the TSS segment selector for the new task as the operand of the JMP or CALL instruction, from a task gate, or from the previous task link field (for a task switch initiated with an IRET instruction).

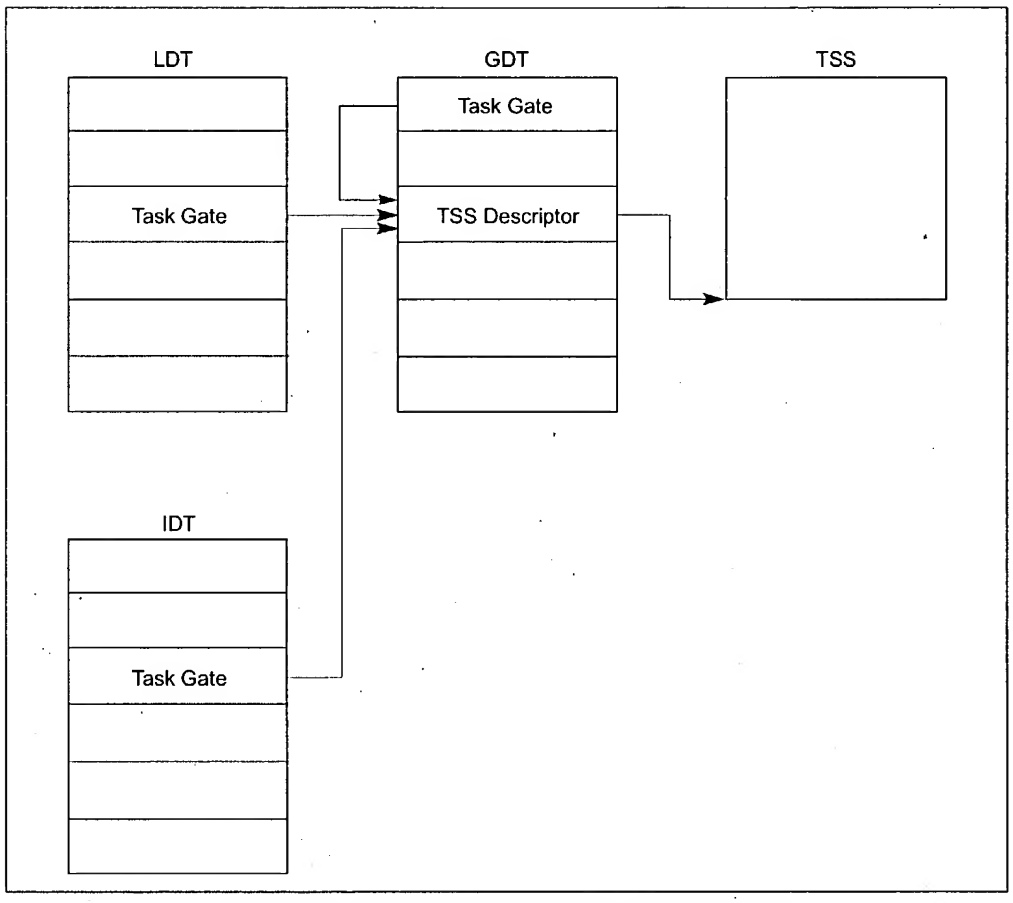

Figure 6-6. Task Gates Referencing the Same Task

2. Checks that the current (old) task is allowed to switch to the new task. Data-access privilege rules apply to JMP and CALL instructions. The CPL of the current (old) task and the RPL of the segment selector for the new task must be less than or equal to the DPL of the TSS descriptor or task gate being referenced. Exceptions, interrupts (except for interrupts generated by the INT *n* instruction), and the IRET instruction are permitted to switch tasks regardless of the DPL of the destination task-gate or TSS descriptor. For interrupts generated by the INT *n* instruction, the DPL is checked.
3. Checks that the TSS descriptor of the new task is marked present and has a valid limit (greater than or equal to 67H).
4. Checks that the new task is available (call, jump, exception, or interrupt) or busy (IRET return).

5. Checks that the current (old) TSS, new TSS, and all segment descriptors used in the task switch are paged into system memory.
6. If the task switch was initiated with a JMP or IRET instruction, the processor clears the busy (B) flag in the current (old) task's TSS descriptor; if initiated with a CALL instruction, an exception, or an interrupt, the busy (B) flag is left set. (Refer to Table 6-2.)
7. If the task switch was initiated with an IRET instruction, the processor clears the NT flag in a temporarily saved image of the EFLAGS register; if initiated with a CALL or JMP instruction, an exception, or an interrupt, the NT flag is left unchanged in the saved EFLAGS image.
8. Saves the state of the current (old) task in the current task's TSS. The processor finds the base address of the current TSS in the task register and then copies the states of the following registers into the current TSS: all the general-purpose registers, segment selectors from the segment registers, the temporarily saved image of the EFLAGS register, and the instruction pointer register (EIP).

NOTE

At this point, if all checks and saves have been carried out successfully, the processor commits to the task switch. If an unrecoverable error occurs in steps 1 through 8, the processor does not complete the task switch and insures that the processor is returned to its state prior to the execution of the instruction that initiated the task switch. If an unrecoverable error occurs after the commit point (in steps 9 through 14), the processor completes the task switch (without performing additional access and segment availability checks) and generates the appropriate exception prior to beginning execution of the new task. If exceptions occur after the commit point, the exception handler must finish the task switch itself before allowing the processor to begin executing the task. Refer to Chapter 5, *Interrupt and Exception Handling* for more information about the affect of exceptions on a task when they occur after the commit point of a task switch.

9. If the task switch was initiated with a CALL instruction, an exception, or an interrupt, the processor sets the NT flag in the EFLAGS image stored in the new task's TSS; if initiated with an IRET instruction, the processor restores the NT flag from the EFLAGS image stored on the stack. If initiated with a JMP instruction, the NT flag is left unchanged. (Refer to Table 6-2.)
10. If the task switch was initiated with a CALL instruction, JMP instruction, an exception, or an interrupt, the processor sets the busy (B) flag in the new task's TSS descriptor; if initiated with an IRET instruction, the busy (B) flag is left set.
11. Sets the TS flag in the control register CR0 image stored in the new task's TSS.
12. Loads the task register with the segment selector and descriptor for the new task's TSS.

- The processor supports a nominal 1-MByte physical address space (refer to Section 16.1.1., "Address Translation in Real-Address Mode" for specific details). This address space is divided into segments, each of which can be up to 64 KBytes in length. The base of a segment is specified with a 16-bit segment selector, which is zero extended to form a 20-bit offset from address 0 in the address space. An operand within a segment is addressed with a 16-bit offset from the base of the segment. A physical address is thus formed by adding the offset to the 20-bit segment base (refer to Section 16.1.1., "Address Translation in Real-Address Mode").
- All operands in "native 8086 code" are 8-bit or 16-bit values. (Operand size override prefixes can be used to access 32-bit operands.)
- Eight 16-bit general-purpose registers are provided: AX, BX, CX, DX, SP, BP, SI, and DI. The extended 32 bit registers (EAX, EBX, ECX, EDX, ESP, EBP, ESI, and EDI) are accessible to programs that explicitly perform a size override operation.
- Four segment registers are provided: CS, DS, SS, and ES. (The FS and GS registers are accessible to programs that explicitly access them.) The CS register contains the segment selector for the code segment; the DS and ES registers contain segment selectors for data segments; and the SS register contains the segment selector for the stack segment.
- The 8086 16-bit instruction pointer (IP) is mapped to the lower 16-bits of the EIP register. Note this register is a 32-bit register and unintentional address wrapping may occur.
- The 16-bit FLAGS register contains status and control flags. (This register is mapped to the 16 least significant bits of the 32-bit EFLAGS register.)
- All of the Intel 8086 instructions are supported (refer to Section 16.1.3., "Instructions Supported in Real-Address Mode").
- A single, 16-bit-wide stack is provided for handling procedure calls and invocations of interrupt and exception handlers. This stack is contained in the stack segment identified with the SS register. The SP (stack pointer) register contains an offset into the stack segment. The stack grows down (toward lower segment offsets) from the stack pointer. The BP (base pointer) register also contains an offset into the stack segment that can be used as a pointer to a parameter list. When a CALL instruction is executed, the processor pushes the current instruction pointer (the 16 least-significant bits of the EIP register and, on far calls, the current value of the CS register) onto the stack. On a return, initiated with a RET instruction, the processor pops the saved instruction pointer from the stack into the EIP register (and CS register on far returns). When an implicit call to an interrupt or exception handler is executed, the processor pushes the EIP, CS, and EFLAGS (low-order 16-bits only) registers onto the stack. On a return from an interrupt or exception handler, initiated with an IRET instruction, the processor pops the saved instruction pointer and EFLAGS image from the stack into the EIP, CS, and EFLAGS registers.
- A single interrupt table, called the "interrupt vector table" or "interrupt table," is provided for handling interrupts and exceptions (refer to Figure 16-2). The interrupt table (which has 4-byte entries) takes the place of the interrupt descriptor table (IDT, with 8-byte entries) used when handling protected-mode interrupts and exceptions. Interrupt and exception vector numbers provide an index to entries in the interrupt table. Each entry provides a pointer (called a "vector") to an interrupt- or exception-handling procedure. Refer to

Table A-2. Events That Can Be Counted with the Pentium® Processor Performance-Monitoring Counters (Contd.)

Event Num.	Mnemonic Event Name	Description	Comments
0FH	ANY SEGMENT REGISTER LOADED	Number of writes into any segment register in real or protected mode including the LDTR, GDTR, IDTR, and TR.	Segment loads are caused by explicit segment register load instructions, far control transfers, and task switches. Far control transfers and task switches causing a privilege level change will signal this event twice. Note that interrupts and exceptions may initiate a far control transfer.
10H	Reserved		
11H	Reserved		
12H	Branches	Number of taken and not taken branches, including conditional branches, jumps, calls, returns, software interrupts, and interrupt returns.	Also counted as taken branches are serializing instructions, VERR and VERW instructions, some segment descriptor loads, hardware interrupts (including FLUSH#), and programmatic exceptions that invoke a trap or fault handler. The pipe is not necessarily flushed. The number of branches actually executed is measured, not the number of predicted branches.
13H	BTB_HITS	Number of BTB hits that occur.	Hits are counted only for those instructions that are actually executed.
14H	TAKEN_BRANCH_OR_BTB_HIT	Number of taken branches or BTB hits that occur.	This event type is a logical OR of taken branches and BTB hits. It represents an event that may cause a hit in the BTB. Specifically, it is either a candidate for a space in the BTB or it is already in the BTB.
15H	PIPELINE FLUSHES	Number of pipeline flushes that occur. Pipeline flushes are caused by BTB misses on taken branches, mispredictions, exceptions, interrupts, and some segment descriptor loads.	The counter will not be incremented for serializing instructions (serializing instructions cause the prefetch queue to be flushed but will not trigger the Pipeline Flushed event counter) and software interrupts (software interrupts do not flush the pipeline).